Due at the start of class Thursday, April 26, 2012.

**Problem 1.** Use the dynamic programming algorithm to find by hand an optimal parenthesization for multiplying matrices of dimensions are given by the sequence

\[ < 6, 3, 10, 5, 8, 4, 20 > . \]

Show the table. You may use a calculator.

**Problem 2.** The traditional world chess championship is a match of 24 games. The current champion retains the title in case the match is a tie. Not only are there wins and losses, but some games end in a draw (tie). Wins count as 1, losses as 0, and draws as 1/2. The players take turns playing white and black. White moves first, which is an advantage. Assume the champion is white in the first game, has probabilities \( w_w, w_d, \) and \( w_l \) of winning, drawing, and losing playing white, and has probabilities \( b_w, b_d, \) and \( b_l \) of winning, drawing, and losing playing black.

(a) Write down a recurrence for the chance that the champion retains the title. Assume that there are \( g \) games left to play in the match and that the champion needs to win \( i \) games (where \( i \) is either an integer or an integer plus 1/2).

(b) Write a recursive algorithm to compute the chance that the champion retains the title, using the above recurrence.

(c) Produce a memoized version of your algorithm.

(d) Give a dynamic programming algorithm to compute the chance that the champion retains the title.

(e) Analyze the running time of your dynamic programming algorithm.

**Problem 3.** In the Euclidean Traveling-Salesman Tour the cities are points in the Euclidean plane and distances are measured in the standard way. The problem is NP-complete. A Bitonic Euclidean Traveling-Salesman Tour starts at the leftmost city, visits cities from left-to-right until it gets to the rightmost city, and then visits cities from right-to-left until it gets back to the leftmost city. (Of course, each city is visited only once, either going left-to-right or right-to-left.) Use dynamic programming to find an optimal bitonic tour in time \( \theta(n^2) \). Make sure to state your recurrence. HINT: Scan left-to-right keeping track of the optimal left-to-right tour and the optimal right-to-left tour at the same time.